International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 6 (2014) pp. 571-581

A Methodology for Designing Robust Central/Self-Organising Multi-Agent Systems

Yaser Chaaban

Leibniz University of Hanover, Institute of Systems Engineering – System and Computer Architecture,
Appelstrasse 4, Hanover 30167, Germany
chaaban@sra.uni-hannover.de

***Abstract*: Organic Computing (OC) has the objective to use principles that are detected in natural systems. Consequently, OC tries to develop systems that are adaptive, flexible and robust at the same time utilising advantage of the organic properties of OC. In this regard, the robustness of OC systems is a key property, because the environments of such systems are dynamic. In this paper, we propose an interdisciplinary methodology, "Robust Multi-Agent System" (RobustMAS) to characterise robustness of multi-agent systems. RobustMAS allows building robust multi-agent systems in presence of disturbances using the OC concept. It uses a hybrid approach (a combination of central and self-organising form) that is robust enough against disturbances. In this way, RobustMAS guarantees an acceptable system performance by limiting the degradation of the performance in the presence of disturbances. In other words, RobustMAS combines the use of a central Observer/Controller (O/C) architecture, autonomous agents, disturbances and deviations from the planned behaviour aiming to solve coordination problems in multi-agent systems. In this context, RobustMAS solves the conflict between a central controller (i.e., a coordination algorithm) and the autonomy of agents so that the system robustness can be achieved. More accurately, RobustMAS introduces a hybrid coordination of a multi-agent system. This hybrid coordination takes place in three steps: path planning, observation, controlling. Simultaneously, we introduce a metric for the quantitative determination of the robustness.**

***Keywords*:** Organic Computing, Robustness, Hybrid Coordination, Multi-Agent Systems**.**

I. Introduction

The Organic Computing Initiative [5] aims to build flexible, adaptive, and robust systems. Thus, it investigates robustness of distributed self-organising systems. This robustness demonstrates a crucial property of OC systems. As a result, robust systems have the capability to continue working in spite of disturbances so that their major tasks can be carried out.

Robustness of a system can be defined in very diverse ways according to the context. Effective control mechanisms for modern systems are desired in order to attain such systems with a better performance and higher robustness. It is very familiar that robustness will be considered with respect to disturbances. The disturbances affect the robustness of the system and may lead to the suspension of the system in the worst case or may constrain, at least, the functionality of the system (the system works but with a reduced degree of robustness). Therefore, variations of the disturbance size are needed in order to study the degree of the system robustness. The disturbance size affects the length of the recovery phase which is required by the system to work robustly again. Briefly, if a system is provided with self-healing properties, this system will be robust against failures or disturbances which may occur.

Because environments of complex systems may change dynamically, self-organising systems should be provided with some degrees of autonomy so that they can adapt their behaviour to new environmental situations. This autonomy as well as disturbances and other reasons may cause an unwanted emergent behaviour [6] or the whole system may fail unexpectedly. Therefore, the system should be observed (e.g., by an observer) and controlled (e.g., by a controller) so that this emergent behaviour or the complete system failure can be prevented. Consequently, the system performance remains effective and will not deteriorate significantly or at least the system will not fail completely.

The main point here is that using a fully centralised approach to design systems is not sufficiently robust, because this design form has a single point of failure. On the contrary, a decentralised approach exhibits more robustness than a centralised approach in many situations; however it often requires overhead costs (e.g., a high overhead in terms of communication). In accordance to this, a hybrid approach including both centralised and decentralised elements will provide a certain degree of robustness, which is one of the main issues of this paper.

It is noteworthy that the definition of system robustness varies according to the context in which the system is used. Therefore, manifold meanings of system robustness were introduced in literature. Additionally, various formal measures and metrics were presented to achieve the system robustness.

Although there are numerous research projects made towards building robust multi-agent systems in diverse fields, a study

of robustness of technical systems, which are modelled as multi-agent systems, does not exist yet (at least it is extremely rare, e.g., an attempt by the Organic Computing Initiative [5]).

This paper is organised as follows. Section 2 explains the application scenario used in this work, a traffic intersection without traffic lights. This scenario serves as a testbed for the evaluation of the RobustMAS concept. Section 3 presents a survey of related work concerning architectures applied to various technical systems. In Section 4 the concept and objectives of RobustMAS will be presented. Furthermore, the problem domain, the components, the agent classes and the proposed system architecture of RobustMAS will be studied. Afterwards, the measurement of robustness of a multi-agent system according to the RobustMAS concept will be discussed, where a new method for their measurement has been developed accordingly. Finally, Section 4 will summarise the presented RobustMAS concept and gives a peek at future trends.

II. AN APPLICATION SCENARIO FOR ROBUSTMAS CONCEPT

This section describes the application scenario of the RobustMAS concept, a traffic intersection without physical traffic lights. In this scenario, autonomous vehicles attempt to cross the intersection as fast as possible.

For this reason, an intersection control algorithm based on virtual traffic lights is used. Such scenarios contain and assemble the required concerns that can be used to build robust multi-agent systems. In this context, autonomous agents are autonomous vehicles, and the controller of the intersection is the central unit. However, the basic idea of the RobustMAS concept is applicable for other systems as well.

In this scenario, a resource sharing problem (resource sharing conflict) arises, which has to be resolved in order to avoid collisions within the intersection (a shared resource). Thus, the coordination of autonomous vehicles is the problem of this application scenario, which will be used later for the evaluation of the RobustMAS concept. A trajectory-based approach will be used where dynamic replanning of trajectories will be investigated in the presence of disturbances.

In this regard, a special problem domain of RobustMAS has been defined making use of the traffic problem as an application scenario for RobustMAS. This domain, which is called "RobustMAS Traffic", deals with intersections of autonomous vehicles in order to solve the traffic problem.

Similar to the RobustMAS concept, the terms of the special application domain, RobustMAS Traffic, can be utilised. For this purpose, the words agent, which is used in RobustMAS, and vehicle, which is used in RobustMAS Traffic, can be used interchangeably. Additionally, the term “shared environment” in RobustMAS is used interchangeably for “centre of the intersection” in RobustMAS Traffic.

In addition to that, the autonomous vehicles and the environment, an intersection without traffic lights, should be observed. This observation aims to detect deviations from plan (trajectories of vehicles) or disturbances (accidents) that may occur. Consequently, replanning and corrective intervention will be directed, if necessary, toward replanning (trajectories replanning) so that the system remains demonstrating safety and robustness.

Figure 1 illustrates the form of the traffic intersection without traffic lights. Here, the intersection was modelled as a grid-based layout. Vehicles that are controlled by agents, try to move through the intersection as quickly as possible.

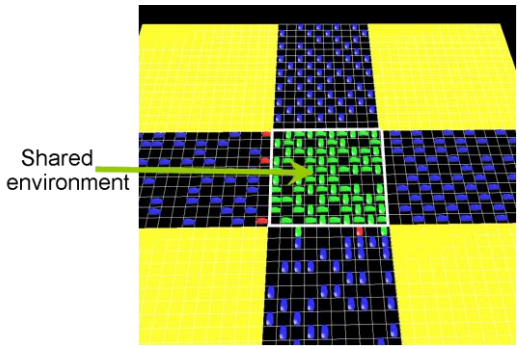

Figure 1. The intersection without traffic lights “RobustMAS Traffic”

Vehicles behave differently regarding their locations, outside or inside the centre of the intersection (shared environment). Vehicles, which are outside the shared environment, attempt to move forward avoiding collisions (act in a fully autonomous way). However, vehicles get collision-free trajectories from the central controller of the intersection. These planned trajectories are provided to vehicles as a recommendation, so that every vehicle has its best possible (desired from controller) path inside the centre of the intersection. Therefore, autonomous vehicles either move faster than their planned trajectories causing deviations from the planned behaviour, or they follow them if deviations are not possible. Here, it is worth mentioning the assumption that the wishes for turning of vehicles are known.

In this application scenario, RobustMAS aims to develop a robust traffic intersection, in the presence of accidents (disturbances) in the intersection, and unplanned autonomous behaviour of vehicles (deviations from planned trajectories). In this regard, the robustness measurement is based on the size of the accident (disturbance strength). Therefore, the simulation has been carried out in the cases that the size of the accident is 1, 2, and 4 (the accident occupies an area of size 1, 2 and 4 cells inside the intersection).

For generalisation of the RobustMAS concept, the current scenario used in this work, intersections without traffic lights, can be replaced also with the more general scenario, shared spaces. Shared space is an approach, developed by Hans Monderman [7]. This generalisation may be possible due to the similarities between the working circumstances and the environments presented in both systems. In this regard, both systems can be considered as unregulated traffic space, where vehicles move in a fully autonomous way without traffic lights.

In previous papers, we introduced a system for coordinating

vehicles at a traffic intersection using an o/c architecture [1][2]. Additionally, handling of deviations from planned (desired) behaviour was studied in [3], whereas handling of disturbances (accidents) was considered in [4].

III. STATE OF THE ART

As mentioned above, in the literature, there are enormous works concerning the robustness of systems. However, there is a clear lack of study of robustness, to the best of our knowledge, in developing robust multi-agent systems in technical systems.

In the literature, diverse architectures were suggested in order to be applied to various technical systems. Architectures for technical systems are depending on specific requirements using design principles and methodologies in order to achieve desired goals, to solve specific problems, to create behaviour patterns of the technical system applied to.

The Adaptive Agent Architecture (AAA) introduced in [8][9] is a multi-agent system architecture that was developed on the basis of the research in fault-tolerance and agent communication languages. This architecture works closely with the Open Agent Architecture. It was employed in multi-agent systems like Quickset [9]. Additionally, it depends on the teamwork-based approach, which is a decentralised approach. Due to the fault-tolerant trait of AAA-architecture, a robust multi-agent system can be designed by means of this architecture.

The AAA architecture is not useful for RobustMAS, because of its approach, which assumes that agents work as teams. This approach does not comply with the RobustMAS concept, which supposes that the agents are self-interested.

Other work relating to the architectures proposed in order to solve collaborative or coordinate problems in multi-agent systems can be summarised as follows:

- An application of the generic O/C architecture was presented in [6]. This application was applied to swarm robot scenarios, where the observer determines the unwanted clustering behaviour of robots. However, this application addresses only clustering behaviour, while RobustMAS deals with disturbances and deviations from plan (desired behaviour).
- A computational framework for the coordination of large robot teams (at least 100 robots) was developed and implemented in the CentiBOTS project [10]. As a result, the CentiBOTS project does not deal with turbulent environment (disturbances).
- A novel modelling methodology for distributed and collectively intelligent systems was proposed in [11]. The resultant methodology does not consider the system robustness against disturbances.
- The Centre for Robot-Assisted Search and Rescue at the University of South Florida has extended the Sensor Fusion Effects (SFX) architecture that serves as the base for the cognitive model of a team of robots. The aim of this extension was to insert a distributed layer so that the concept of a person from psychology can be mimicked. This architecture is called the Distributed Field Robot Architecture (DFRA) [12]. However, DFRA architecture does not take into account the influence of disturbances on system functionality, while RobustMAS tries to reduce the effect of disturbances on system performance.
- A behavioural architecture for swarm robots was suggested in [13]. This architecture is very effective for self-assembling tasks (swarm of self-assembling robots). In this architecture, the key role is played by the interactions among agents, which are responsible for the formation of the needed pattern. On the contrary, RobustMAS uses a central component that performs the desired behaviour (collision-free trajectories), where this planned behaviour is given to agents only as a recommendation.
- The Autonomic Nano-Technology Swarm (ANTS) is a generic mission architecture introduced by NASA. The goal of NASA is to utilise approaches of multi-agent systems in space missions. The ANTS architecture is a mission/system architecture that can be applied to robust, scalable, highly distributed systems [14]. However, ANTS architecture has no consideration for the system robustness when disturbances occur in the environment.
- Different distribution possibilities of the generic O/C architecture were investigated in [15]. The study aims to create collaboration patterns in multi-agent systems using the O/C architecture and to apply it to a traffic scenario.

Briefly, it can be seen that most system architectures discussed above are focused on specific problems aiming to solve them (collaborative or coordinate problems) in context of multi-agent systems. However, the generic O/C architecture presented in [6] introduces generic methodologies and approaches, where the observation and control of such systems will supply the desired results avoiding unwanted behaviour of agents. In this regard, RobustMAS uses an O/C architecture to observe autonomous agents within a shared environment in order to detect deviations (unplanned autonomous behaviours) from desired behaviour. Additionally, RobustMAS intervenes when it is necessary, so that the system maintains a desired level of system performance in spite of disturbances in the environment. Consequently, RobustMAS focuses on the robustness of hybrid central/self-organising multi-agent systems.

In previous paper [4], we focused the discussion of related work on robust agent-based approaches used for fully autonomous vehicles within an intersection without traffic lights. In this context, a study of the impact of a multi-agent intersection control protocol for fully autonomous vehicles on driver safety is presented in [16]. In this study, the simulations deal only with collisions in intersections of autonomous vehicles aiming to minimise the losses and to mitigate catastrophic events. However, it can be noted that the study has not considered the robustness of the intersection system.

Furthermore, we considered various methods for measuring robustness [4]. To address this issue, some approaches were introduced, among others, in [17][18][19]. Both approaches;

the FePIA procedure in [17] and the statistical approach in [18] are general approaches and consequently can be adapted to specific purposes (arbitrary environment). In both approaches, diverse general metrics were used to quantify robustness.

There is also much other related work that can be found in the literature, e.g., on re-planning, plan repair, formal analysis of protocols for emergent behaviours, and so on. Finally, sensor networks can be considered as MAS and there is much research published on robustness and fault-tolerance in sensor networks (see [20] [21] for examples). Here, fault tolerance is one of the critical issues in wireless sensor networks (WSNs).

IV. THE ROBUSTMAS APPROACH

The concept and objectives of RobustMAS will be presented in the next sections. Additionally, the problem domain, the components, the agent classes and the proposed system architecture of RobustMAS will be clarified highlighting the hybrid central/self-organising architecture as the key concept of RobustMAS. Subsequently, the measurement of robustness and gain of a multi-agent system according to the RobustMAS concept will be presented in term of definition and proposition of a new appropriate method for their measurement.

As mentioned above, for the explanation of the RobustMAS concept, the words agent and vehicle are used interchangeably. Also, the term "shared environment" is used interchangeably for "centre of the intersection". Finally, the term "disturbance" is used interchangeably for "accident", and the term "desired behaviour" for "planned trajectories".

A. Robust system with disturbance

The RobustMAS concept introduces a robust hybrid central/self-organising multi-agent system (hybrid coordination) solving the conflict between a central planning algorithm and the autonomy of the agents (decentral, self-organised). Here, the autonomy of the agents is recognised as a deviation from the plan of the central algorithm, if the agents are not respecting this plan.

The application scenario used in this work is an intersection without traffic lights, where vehicles are modelled as autonomous (semi-autonomous) agents (Driver Agents) with limited local capabilities. The vehicles are trying as quickly as possible to cross the intersection without traffic lights. In the meantime, an interaction between decentralised mechanisms (autonomous vehicles) and centralised interventions arises. Here, the goal is to build a robust intersection without traffic lights when disturbances (e.g., accidents) and deviations (e.g., unplanned autonomous behaviour) occur.

Moreover, RobustMAS addresses a further problem that occurs in the system wherever multiple agents (e.g., robots, vehicles, etc.) move in a common environment. This problem is called resource sharing conflict (Resource Allocation Problem). This problem raises the question: "How can agents of a system move reliably in their environment?". RobustMAS uses coordination mechanisms (a manager is responsible for coordinating tasks) to solve the resource sharing conflict. These coordination mechanisms are based on the idea of path planning, which must be performed taking into consideration other agents (vehicles) and the geometry of the environment (intersection). The path planning is performed in a 3-dimensional space with two geometrical dimensions (x, y) representing the intersection and time t.

For the path planning, RobustMAS uses an adapted A*-algorithm to calculate collision-free trajectories (central planning) for all agents in a shared environment (the centre of the intersection) enabling them to avoid collisions. This path planning (collision-free trajectories) is given to agents as a recommendation.

Since the agents are autonomous (decentral, self-organised) and thus deviations from the plan (trajectories) in principle are possible, RobustMAS performs an observation of compliance with these trajectories (e.g., by an observer).

RobustMAS aims to make the system capable to return to its normal state with minimal central planning intervention after disturbances occur (e.g., by a controller).

Robust systems should be fault-tolerant in order to deal with faults, deviations or disturbances and to continue working effectively and fulfilling their major tasks. In the context of this work, fault tolerance avoids system failures in the presence of deviations and disturbances that occur in the system allowing the agents of the system to move reliably in their environment.

In order to conceive the basic idea of RobustMAS, three cases of the system operation will be considered:

1. Operation without disturbance.
2. Operation with disturbance without intervention.
3. Operation with disturbance with intelligent intervention.

Figure 2 illustrates the main idea of this concept in establishing a robust system that tolerates faults, disturbances and deviations which could be occurred in the system.

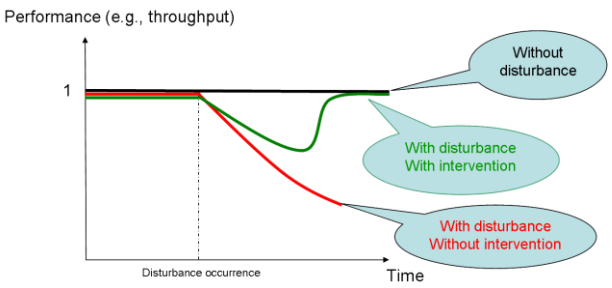

Figure 2. Robust system with disturbance occurrence

As depicted in Figure 2, the performance (e.g., throughput) of the system is at its best (i.e., equal to 1) when no disturbances occur. When a disturbance occurs, the system performance would begin to fall and probably it would become worse (deteriorate) over time, if no corrective intervention is taken in due time. In contrast, if the corrective intervention is intelligent and fast enough, the system performance should improve in the course of time when a disturbance occurs. This means that the system performance remains acceptable despite the occurrence of disturbance.

B. Goals (contributions) of RobustMAS

The main contribution of this work is the integration of concepts from different research areas into a practically applicable methodology. Figure 3 summarises the methodologies integrated within RobustMAS.

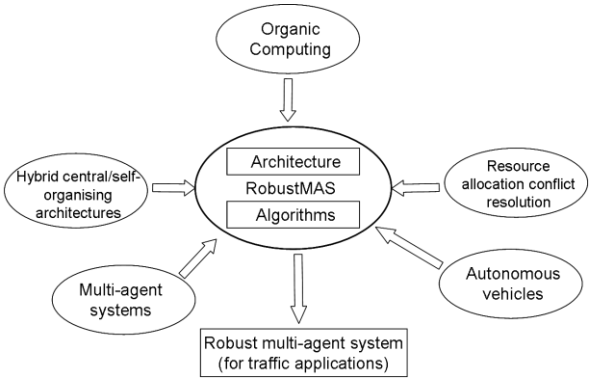

Figure 3. The methodologies integrated within (RobustMAS)

The main goal of the new concept (RobustMAS) is to solve the conflict between a central planning algorithm and the autonomy of the agents using a hybrid form of a central/self-organising solution of the coordination problem for multi-agent systems. This approach:

- Keeps a multi-agent system at a desired performance level when disturbances and deviations occur.
- Coordinates autonomous/semi-autonomous agents.
- Recognises the autonomy of the agents as a deviation from the plan.
- Tolerates that some agents behave in a fully autonomous way.
- Tolerates that some autonomous agents leave the control of the fully central architecture.
- Forms a hybrid central/self-organising architecture for a multi-agent system, which is a special form of the fully central architecture.
- Deals with turbulent environment (disturbances).
- Has the goal to develop a robust multi-agent system despite disturbances and deviations in the system (internal) or in the environment (external).

A key point in the work is the coordination of autonomous vehicles. This is the central component of the application example, a traffic intersection without traffic lights, which will be used for the evaluation.

Furthermore, RobustMAS establishes a robust traffic intersection without traffic lights. Here, the deviations will be first detected by the observer, so that the controller could intervene in time, if needed, in order to guarantee the robustness of the intersection. A disturbance is, for example, an accident in the intersection; and a deviation is, for example, an unplanned autonomous behaviour of a vehicle.

In addition, RobustMAS solves a coordination problem by a central algorithm (a central-planning algorithm), using an adapted A*- algorithm that was used for path planning. Here, the path planning is considered as a resource allocation problem (resource sharing problem) where multiple agents move in a shared environment and need to avoid collisions. For evaluation, it is necessary to determine the degree of the system robustness using a suitable metric, which quantifies this robustness.

C. Hybrid central/self-organising concept for multi-agent systems

In this work, the term "hybrid central/self-organising multi-agent system" is introduced. It is a new possibility of the distribution of the proposed architecture.

Figure 4 shows the main idea of this hybrid central/self-organising concept derived from the fully central architecture.

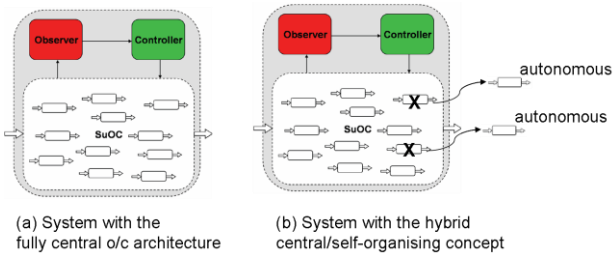

Figure 4. The hybrid central/self-organising concept

(a) Fully central architecture: One O/C for the whole system under observation and control.

(b) Hybrid central/self-organising concept: One O/C for the whole system under observation and control, but the autonomous agents can leave the control of the fully central architecture to behave in a fully autonomous way (but still under observation).

The hybrid central/self-organising concept aims to increase the autonomy of agents compared to the central architecture. This means, the hybrid concept tolerates that some agents behave autonomously. It solves the conflict between a central planning algorithm (a component in the controller) and the autonomy of the agents (the entities of the system under observation and control). The autonomy of the agents is recognised as a deviation from the plan of the central algorithm, if the agents are not respecting this plan.

Figure 5 shows the general flow plan proposed by RobustMAS to solve the conflict between a central planning algorithm and the autonomy of the agents. A central planning algorithm generates a plan for every agent in the system. Since the agents are autonomous and they behave in a completely autonomous way, they may not obey this central plan. If they comply with the central plan then the system works effectively as planned (no deviations from plan). However, if they do not comply with the central plan then RobustMAS detects this deviations from the plan (e.g., by an observer) in order to arrange an appropriate corrective intervention (e.g., by a controller). It makes also replanning, if necessary, with respect to the new situation.

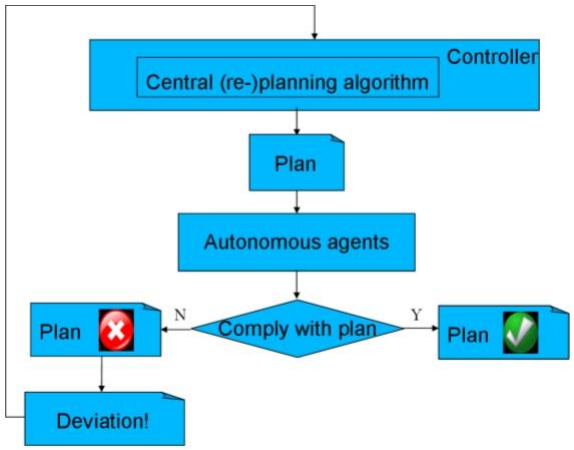

Figure 5. The conflict between a central planning algorithm and the autonomy of the agents (The general flow plan proposed by RobustMAS solving the conflict)

Consequently, RobustMAS comprises the use of a central O/C architecture, autonomous agents and deviations from a central plan in order to solve coordination problems in multi-agent systems. Additionally, it keeps the system at a desired performance level (via replanning and corrective intervention of the controller) when deviations and disturbances occur in the system behaviour, so that the agents of a system can move reliably in their environment.

D. Life cycle of RobustMAS

As mentioned above, the general problem domain of RobustMAS is the resource allocation problem (resource sharing problem) which occurs in the system wherever multiple agents move in a common environment. This section presents the proposed solution to cope with this problem.

RobustMAS uses coordination mechanisms to solve the resource sharing problem. These coordination mechanisms are based on the idea of path planning, which must be performed taking into consideration other agents and the geometry of the shared environment in the configuration space-time (x, y, t). Here, the path planning is considered as a resource allocation problem (resource sharing problem).

Since the goal of RobustMAS is to keep a multi-agent system at a desired performance level when disturbances and deviations occur in the system behaviour, agents have to be observed (through the observer) within the shared environment. This will be made to intervene (through the controller) in time when it is necessary so that the system remains demonstrating robustness and safety properties. The paradigm of the proposed solution consisting in an Observer/Controller architecture can be seen in Figure 6.

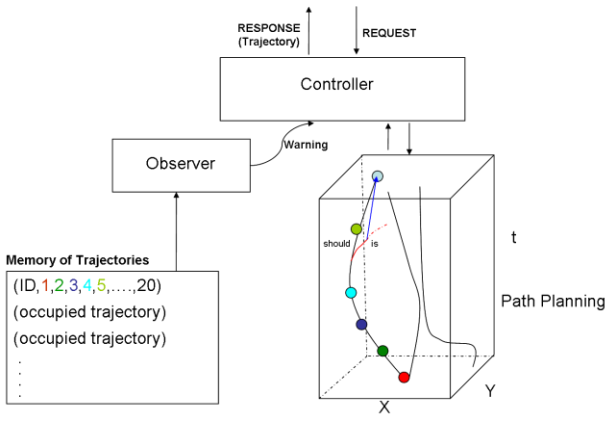

Figure 6. The paradigm of the proposed solution consisting in an Observer/Controller architecture

This Figure depicts a hybrid coordination scheme of a multi-agent system. It takes place in three steps:

1. **Path planning**: The agents send requests to the controller, which computes collision-free trajectories and arranges the participants. This means that the first step is a central planning of the trajectories without deviations of the agents. The agents get their planned trajectories only as recommendation from the controller. Autonomous behaviour of the agents means that they either obey the plan or deviate from it or the agents are completely outside of the plan.
2. **Observation**: The observation of actual trajectories of agents in the shared environment is done by an observer component in order to identify eventual deviations from the plan, using the memory of all planned trajectories. The observer informs the controller about its observation.
3. **Controlling**: The controller carries out a replanning for the trajectories of the affected agents, if needed, in order to accomplish an appropriate corrective intervention. The controller uses a decision mechanism to take a decision how it could intervene most suitably.

E. Agent classes

In this section, the agent classes created in order to be used by RobustMAS will be described.

Each agent class represents a specialised role that can be performed by the agents of this class in run time. Each class has certain capabilities in order to interact with other agent classes, which should take into account the whole goal of the desired system.

RobustMAS implements agent classes allowing the agents to play their roles. Based on the type of their class that they belong to, the agents try to maximise:

- **Class 1:** Only their own fitness (e.g., their own utility), which can be achieved by travelling across the environment as quickly as possible, i.e., minimisation of their individual travel times of agents across the environment, or
- **Class 2:** Only the fitness of the whole system (the system throughput), or
- **Class 3:** Their own fitness and then the fitness of the whole system respectively in every step.

These agents are either Non-Autonomous Agents (NAA) or Autonomous Agents (AA).

$$A = \{NAA, AA\} \quad (1)$$

In turn, Autonomous Agents (AA) are either Autonomous and Rational Agents (ARA) or Autonomous and Non-Rational Agents (ANRA).

$$AA = \{ARA, ANRA\} \quad (2)$$

In this regard, "rational" means "reasonable autonomy", i.e., agents are aware of their capabilities to make a rational choice of an action that is reasonable to maximise their own utility. However, and simultaneously, these agents follow safety rules carefully, so that they do not cause resource sharing conflicts (efficiently aware of their environment).

As a result, these agents by RobustMAS are generally classified as follows:

- **Class 1:** Autonomous and Non-Rational Agents (ANRA): They deviate from the plan and cause disturbances. These agents are competitive. They try to maximise only their own fitness (e.g., their own utility) and they do not consider the fitness of the whole system (e.g., the system throughput). However, they do not agree to the allocated resources and they cause possibly a resource sharing conflicts with other agents, because of their non-rationality.
- **Class 2:** Non-Autonomous Agents (NAA): They do not deviate from the plan and do not cause disturbances. These agents are cooperative. They try indirectly to maximise the fitness of the whole system (e.g., the system throughput) and they agree to the allocated resources. That means they do not cause resource sharing conflicts.
- **Class 3:** Autonomous and Rational Agents (ARA): They deviate from the plan, but do not cause disturbances. These agents are cooperative and competitive at the same time. They try to maximise their own fitness (e.g., their own utility) and then the fitness of the whole system (e.g., the system throughput). However, they do not agree to the allocated resources, but they do not cause resource sharing conflicts, because of their rationality.

F. System architecture

This section gives an overview of the proposed system architecture and how to implement it on a highly relevant technical problem: the control of autonomous agents moving in a shared environment demonstrating a robust multi-agent system. Additionally, it describes the adaptation of this architecture to the traffic intersection without traffic lights.

Figure 7 shows the detailed internals of the RobustMAS architecture. The system under observation and control is considered as a set of elements possessing certain attributes in terms of multi-agent systems. This system under observation and control contains all agents that move within the shared environment avoiding collisions. The agents outside the shared environment send messages (requests) to the controller which replies with collision-free planned trajectories for all agents (path planning unit).

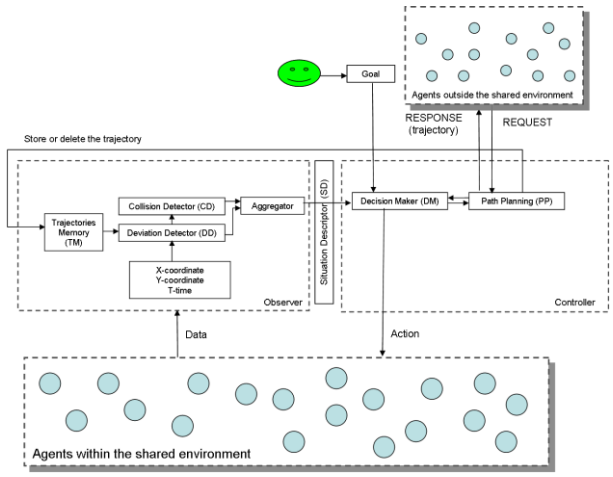

Figure 7. Detailed RobustMAS system architecture

Every agent by itself is assumed to be egoistic (class 1 and 3 agents), because it is autonomous and tries to quickly cross the shared environment so that it may not obey its planned trajectory. Therefore competition situations arise due to the egoistic behaviour (competition-based behaviour) of agents, which in turn leads to congestions, where agents with different moving directions block each other in the common environment. These congestions may cause a large cluster of blocked agents for a long time.

The observer reads the planned trajectory of an agent from the trajectory memory (memory of trajectories unit TM) only when this agent is located within the shared environment and compares it with the agent's actual travelled trajectory using the deviation detector (deviation detector unit DD) to identify all deviations from the planned trajectories. The observer uses also the collision detector (collision detector unit CD) to detect whether a deviation led to a collision and to detect the deviation class (see below). Afterwards, it aggregates (aggregator unit) its observations as a vector of situation parameters (situation descriptor unit SD). These parameters are then sent to the controller. The controller has to intervene on time if necessary (decision maker unit DM) and to select the best corrective action (it makes a decision whether a replanning is required and uses also the path planning unit PP if needed) that corresponds to the current situation so that the system performance remains acceptable and the target performance of the system is maintained. The intervention of the controller (the decision of the decision maker) will be done with respect to the goal given by the user.

G. Definition of deviation and disturbance in RobustMAS

Since the definition of deviation and disturbance varies according to the context condition, it is necessary to define both terms clearly in the context of this work.

According to the RobustMAS concept, the deviation and the disturbance can be defined as follows:

Definition 1: "A deviation is a different behaviour or path or plan from what was initially planned (desired or expected) for an agent. In other words, a deviation is an unplanned autonomous behaviour. Deviations from the plan of the central planning algorithm occur, if the agents are not respecting this plan".

Definition 2: "A disturbance is a permanent change in environmental conditions, which leads to an unwanted evident change in the target performance of the system. Moreover, disturbances are obstacles (blocked surfaces, restricted areas, or any additional difficulty) in the way of the agents. These obstacles block agents in the neighbourhood causing longer delays than planned".

Additionally, the disturbance strength can be defined according to the RobustMAS concept as follows:

Definition 3: "A disturbance strength is a positive constant defining the strength (size) of the disturbance".

H. Measurement of robustness and gain

Since RobustMAS aims to keep a multi-agent system at a desired performance level even though disturbances and deviations occur in the system, a method to measure the robustness of a multi-agent system is required. The equivalent goal of RobustMAS by the application scenario, a traffic intersection without traffic lights, is to keep the traffic intersection at a desired performance level even though deviations from the planned trajectories and accidents occur in the intersection. Therefore, a new concept will be introduced in order to define the robustness of multi-agent systems. Additionally, the gain of RobustMAS will be defined and used to show the benefit of the hybrid central/self-organising concept.

The robustness of a multi-agent system can be defined as follows:

Definition 4: Robustness:

"A (multi-agent) system is considered robust against disturbances if its performance degradation is kept at a minimum".

Consequently, the RobustMAS concept assumes that a robust system keeps its performance acceptable after occurrence of disturbances and deviations from the plan.

Definition 5: Relative robustness:

"The relative robustness of a (multi-agent) system in the presence of a disturbance is the ratio of the performance degradation due to the disturbance divided by the undisturbed performance".

In order to measure the robustness of RobustMAS in the traffic intersection system, the throughput metric is used for determining the reduction of the performance (system throughput) of RobustMAS after disturbances (accidents) and deviations from the planned trajectories. That is because throughput is one of the most commonly used performance metrics. Therefore, the comparison of the throughput values is required in the three cases:

(1) Without disturbance.
(2) With disturbance with intervention.
(3) With disturbance without intervention.

Based on this, the robustness measurement of RobustMAS can be considered in two ways:

- Using cumulative system performance, i.e., cumulative throughput (#Agents), where the system is considered only until the time when the disturbance ends. We introduced this way of robustness measurement in [4].
- Using system performance, i.e., throughput per time unit (#Agents/sec), where the system is considered until the time when the system returns after disturbances to its normal state like before.

For this explanation of the robustness measurement, the words agent and vehicle can be used interchangeably.

I. Measuring robustness using system performance (throughput per time unit):

In this case, the system performance, i.e., throughput per time unit (#Agents/sec) is used. Additionally, the system is considered longer than in the case of the cumulative performance (cumulative throughput) values. Therefore, compared to that case that defines time t_1, the occurrence time of disturbance, and time t_2, the end time of disturbance, the times t_3 and t_4 will also be defined. Here, t_3 is the time at which the system returns to its normal state with minimal central planning intervention, while t_4 is the time at which the system returns to its normal state without central planning intervention. In this regard, the normal state represents the system performance level at its best when no disturbances occur (under normal operating conditions). Here, we use the following functions:

- P_0 (t): represents the system performance when no disturbances occur (normal state).
- $P_{d,\ ni}$ (t): represents the system performance with a disturbance with no intervention by the central planning.
- $P_{d,\ i}$ (t): represents the system performance with a disturbance with an intervention of the central planning.

Figure 8 shows the performance (throughput per time unit) values of the system before and after the disturbance until the time when the system returns to its normal state like before comparing the three mentioned cases.

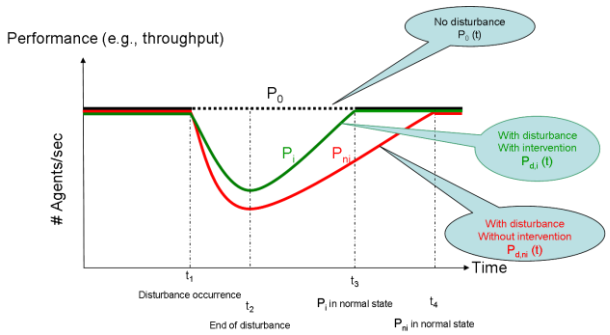

Figure 8. Comparison of system performance (throughput per time unit) for three situations

In accordance with the definition 5 mentioned above, the relative robustness (R) of a system (S) is determined as follows:

$$R = \frac{\int_{t1}^{t4} P_{d,i}(t)\,d(t)}{\int_{t1}^{t4} P_0(t)\,d(t)}\ ;\quad 0 \le R \le 1 \qquad (3)$$

Here, the lower and upper boundaries can be set as follows:

- R = 0 represents the lower boundary case of the relative robustness, where the system is considered as non-robust against disturbances (very poor performance). It appears

when $P_{d,i}(t) << P_0(t)$, i.e., the performance degradation is very strong due to the disturbance in spite of the intervention, compared to the performance when no disturbance occurs. Thus, the system behaviour is not acceptable in the face of disturbances.

- R = 1 represents the upper boundary case of the relative robustness, where the system is considered as strongly robust against disturbances (an optimal performance, an ideal behaviour). It occurs, when $P_{d,i}(t) = P_0(t)$, i.e., there is no performance degradation due to the intervention despite the presence of disturbances.

Furthermore, the system could be also weakly robust if its performance level is acceptable but not optimal in the presence of disturbances. Therefore, the system behaviour is acceptable but not ideal.

The gain of a system is determined as the difference between the performance in both cases, disturbances with and without intervention:

$$Gain(i \rightarrow ni) = \#Agents(i) - \#Agents(ni) = \int_{t1}^{t4} [P_{d,i}(t) - P_{d,ni}(t)]\, d(t) \quad (4)$$

Consequently, the loss of a system is determined as the difference between the performance in both cases, no disturbance and disturbances with intervention:

$$Loss = \int_{t1}^{t4} [P_0(t) - P_{d,i}(t)]\, d(t) \quad (5)$$

The discussion of the robustness measurement using the system throughput metric will be based on the parameter disturbance strength (see the definition above). In the traffic scenario, the disturbance strength represents the size of the accident in the traffic intersection. Accordingly, the robustness measurement was repeated in the cases that the disturbance strength is 1, 2, and 4. That means, the accident occupies an area of size 1, 2 and 4 cells in the traffic intersection as depicted in Figure 9.

V. SUMMARY AND FUTURE WORK

The Organic Computing (OC) initiative [5] aims to build robust, flexible and adaptive technical systems. Future systems shall behave appropriately according to situational needs. But this is not guaranteed in novel systems, which are complex and act in dynamically changing environments. Therefore, the robustness of OC systems is a key property.

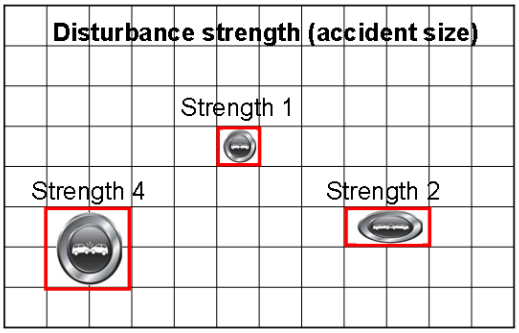

Figure 9. The disturbance strength (the accident size) in three cases: 1, 2, and 4 cells in the traffic intersection

The focus of the interdisciplinary methodology, RobustMAS, is to investigate the robustness of coordination mechanisms for multi-agent systems in the context of OC. RobustMAS poses a challenge to support the multi-agent system with mechanisms to keep the system at a desired performance level when disturbances and deviations from plan occur (robustness). Furthermore, RobustMAS proposes a new appropriate method to measure the robustness of such multi-agent systems.

This work discussed the RobustMAS methodology, followed by a detailed explanation of concept, objectives, agent classes and the proposed architecture and its components. The resulting concept allows building robust multi-agent systems in presence of disturbances.

RobustMAS represents a robust hybrid central/self-organising multi-agent system, in which the conflict between centralised interventions (central planning, a coordination algorithm) and the autonomy of the agents (decentralised mechanisms, autonomous vehicles) was solved.

Furthermore, this work presented the general problem domain of RobustMAS, the resource sharing problem (resource allocation problem), followed by the proposed solution to cope with it. This problem appears in a multi-agent system wherever multiple agents move in a shared environment. In this context, agents struggle to get resources (the shared environment) in order to move over it quickly. Therefore, RobustMAS provides a coordination mechanism to prevent a potential resource sharing conflict. This mechanism uses a concept of path planning so that the required resource allocation is planned over time. Accordingly, the resource allocation is made by a central controller, while the agents employ these resources. The resource planning is done in the configuration space-time (x, y, t), so that the agents can move reliably in their environment.

On the other hand, this work proposed “RobustMAS Traffic” as a special problem domain of the RobustMAS concept. “RobustMAS Traffic” focuses on the traffic problem in an intersection without physical traffic lights. Here, vehicles are modelled as autonomous (semi-autonomous) agents with limited local capabilities. These vehicles try as quickly as

possible to cross the intersection. “RobustMAS Traffic” aims to design a robust traffic intersection in the presence of disturbances (e.g., accidents) and deviations (e.g., unplanned autonomous behaviour). Nonetheless, the concept of RobustMAS is applicable for other systems too.

Finally, the measurement of robustness and gain of a multi-agent system was presented in this work. Subsequently, a method to measure robustness and gain of multi-agent systems was proposed.

In this paper, we presented the RobustMAS concept. Therefore, the next step is to continue with explanation of the realisation of this concept investigating which techniques can be applied to accomplish the three steps of the RobustMAS concept: path planning, observation, and controlling.

Additionally, one aspect that may be of interest for future work is the coordination and cooperation of multiple intersections without traffic lights.

References

[1] Y. Chaaban, J. Hähner, and C. Müller-Schloer, “Towards fault-tolerant robust self-organizing multi-agent systems in intersections without traffic lights”. In *Proceedings of The First International Conference on Advanced Cognitive Technologies and Applications (Cognitive09)*, pp. 467-475, November 2009, Greece. IEEE.

[2] Y. Chaaban, J. Hähner, and C. Müller-Schloer, “Towards Robust Hybrid Central/Self-organizing Multi-agent Systems”. In *Proceedings of the Second International Conference on Agents and Artificial Intelligence (ICAART2010)*, Volume 2 , pp. 341-346, January 2010, Spain.

[3] Y. Chaaban, J. Hähner, and C. Müller-Schloer, “Handling of Deviations from Desired Behaviour in Hybrid Central/Self-Organising Multi-Agent Systems”. In *Proceedings of the Fourth International Conference on Advanced Cognitive Technologies and Applications (Cognitive12)*, pp. 122-128, July 2012, France.

[4] Y. Chaaban, J. Hähner, and C. Müller-Schloer, “Measuring Robustness in Hybrid Central/Self-Organising Multi-Agent Systems”. In *Proceedings of the Fourth International Conference on Advanced Cognitive Technologies & Applications (Cognitive12)*, pp. 133-138, July 2012, France.

[5] CAS-wiki: Organic Computing, http://wiki.cas-group.net/index.php?title=Organic_Computing, [retrieved: January, 2013].

[6] M. Mnif, U. Richter, J. Branke, H. Schmeck, and C. Müller-Schloer, “Measurement and Control of Self-organised Behaviour in Robot Swarms”. *Proceedings of the International Conference on Architecture of Computing Systems (ARCS 2007)*, pp. 209-223, 2007.

[7] Wikipedia: Shared space, http://en.wikipedia.org/wiki/Shared_space, [retrieved: January, 2013].

[8] S. Kumar and P. R. Cohen, “Towards a Fault-Tolerant Multi-Agent System Architecture”. In *Proceedings of the fourth international conference on Autonomous agents*, pp. 459-466, 2000. ACM Press publisher.

[9] S. Kumar, P. R. Cohen, and H. J. Levesque, “The Adaptive Agent Architecture: Achieving Fault-Tolerance Using Persistent Broker Teams”. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, Boston MA, USA, July 7-12, 2000.

[10] K. Konolige , C. Ortiz , R. Vincent , A. Agno , B. Limketkai , M. Lewis , L. Briesemeister , D. Fox , J. Ko , B. Stewart , and L. Guibas, "CENTIBOTS Large Scale Robot Teams". *Artificial Intelligence Center, SRI International*, Menlo Park, CA 2003.

[11] N. Correll and A. Martinoli, "Collective inspection of regular structures using a swarm of miniature robots". In *Proc. of the Int. Symp. on Experimental Robotics (ISER)*. Singapore: Springer Tracts for Advanced Robotics (STAR), Vol. 21, pp. 375-385, June 2006.

[12] M. Long, A. Gage, R. Murphy, and K. Valavanis, "Application of the distributed field robot architecture to a simulated demining task". In *Proc. IEEE ICRA*, Barcelona, Spain, pp. 3193-3200, Apr. 2005.

[13] V. Trianni, Th.H. Labella, R. Gross, E. Sahin, M. Dorigo and J.-L. Deneubourg, “Modeling Pattern Formation in a Swarm of Self-Assembling Robots”. *Technical Report TR/IRIDIA/2002-12*, IRIDIA, Université Libre de Bruxelles, Bruxelles, Belgium, May 2002.

[14] S. A. Curtis, M. L. Rilee, W. Truszkowski, and P. E. Clark, "ANTS for the Human Exploration and Development of Space". In *Proceedings of the IEEE 2003 Aerospace Conference*, Volume 1, pp. 1-7, March 2003.

[15] E. Cakar, J. Hähner, and C. Müller-Schloer, “Creating collaboration patterns in multi-agent systems with generic observer/controller architectures”. In *Proceedings of the 2nd International Conference on Autonomic Computing and Communication Systems (Autonomics 2008)*, pp. 1-9, ICST, Brussels, Belgium, 2008.

[16] K. Dresner and P. Stone, “Mitigating catastrophic failure at intersections of autonomous vehicles”. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS 2008)*, pp. 1393–1396, Richland, SC, 2008.

[17] V. Shestak, H. J. Siegel, A. A. Maciejewski, and S. Ali, “The robustness of resource allocations in parallel and distributed computing systems”. In *Proceedings of the International Conference on Architecture of Computing Systems (ARCS 2006)*, pp. 17–30, 2006.

[18] D. England, J. Weissman, and J. Sadagopan, “A new metric for robustness with application to job scheduling”. In *IEEE International Symposium on High Performance Distributed Computing 2005 (HPDC-14)*, Research Triangle Park, NC, July 24-27, 2005.

[19] K. Waldschmidt and M. Damm, “Robustness in SOC Design”. In *Proceedings of the 9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006)*, Volume: Issue: , 0-0, pp. 27-36, 2006.

[20] M. Effatparvar, Y. Matinfard, M. Hosseinzadeh, and M. Dehghan, “Energy Aware Hybrid PUSH-PULL with

Fault Tolerant Approach in Sensor Networks" *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, vol. 4, pp. 522–529, 2012.

[21] A. Mahapatro and P. M. Khilar, "Fault Diagnosis in Body Sensor Networks" *International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM)*, vol. 5, pp. 252–259, 2013.

Author Biographies

Yaser Chaaban was born in 1974 in Aleppo (Syria). He received the B.Sc. degree in computer engineering from the Aleppo University in 1998, and the M.Sc. in computer engineering from the Leibniz University Hannover (LUH), Germany, in 2005. In 2013 he received the Dr.-Ing. degree from the Faculty of Electrical Engineering and Computer Science of the Leibniz University Hannover, Germany. He worked in the area of control management in distributed multi-agent systems. His research focuses on robustness and coordination in the field of organic computing. After gaining experience as an academic researcher at the Institute of Systems Engineering at the Department of System and Computer Architecture at the LUH he is now working as assistant lecturer at the Studienkolleg of the Leibniz University Hannover.